

The chiller model documented in section 4 of the report was coded in C++. The property routines were developed as described in Appendix B of the report. This appendix begins with a description of all the files that are a part of the software. This is followed by a description of how the model is executed from within Matlab. Finally, the process for simulating the more common faults in the system model is described.

### Installation and file/directory structure:

The complete system model is packaged as a compressed (zip) file named *chillersim1p0.zip*. Installation consists of un-compressing this file to a known directory and including this directory and all sub-directories therein into Matlab's search path.

Fig. 1 shows the file and directory structure seen upon uncompressing *chillersim1p0.zip*:

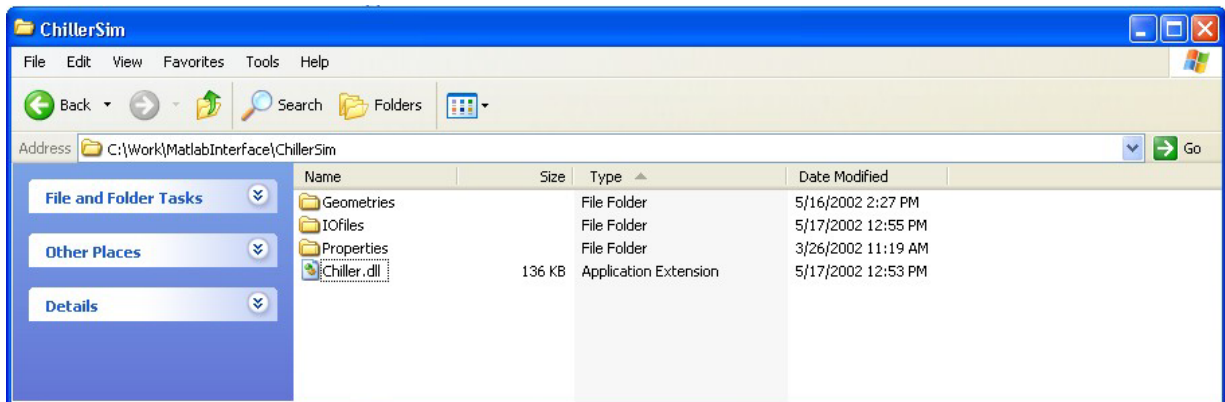


Fig. 1: Screen-shot of unzipped chiller model files

*Chiller.dll* is the dynamically linked library containing all the routines required to run the chiller components and system. The *Geometries* sub-directory contains the text files with the physical constructional details of the various components of the chiller. The *IOFiles* sub-directory contains the text files that are used by the chiller model during initialization and execution. The *Properties* sub-directory contains the property tables that are read into memory when the model is first launched.

Fig. 2 shows the text files in the *Geometries* directory.

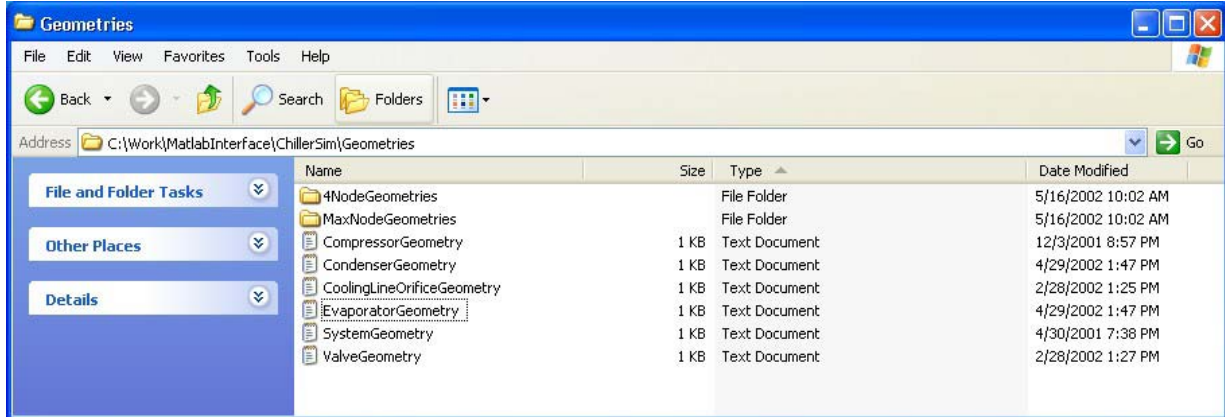


Fig. 2: Screen-shot of the Geometries directory

COMPRESSORGEOMETRY.TXT: This file contains the details required for defining the controller and compressor models. It is strongly recommended that these values not be changed as they could result in unpredictable behavior of the compressor model and hence the system model. Appended at the end of the file is a line-by-line description of parameters.

VALVEGEOMETRY.TXT: This file identifies and lists the constructional parameters used in the valve model. This information consists of (also see Appendix C) the maximum flow-area of the valve, the angle of the valve needle, the discharge coefficient, spring compliance, sensing bulb time constant and the minimum superheat pressure setting.

COOLINGLINEORIFICEGEOMETRY.TXT: This file identifies and lists the required constructional parameters used to define the orifice in the cooling line which is the flow-area of the orifice and the discharge coefficient.

CONDENSERGEOMETRY.TXT, EVAPORATORGEOMETRY.TXT: Since the evaporator and condenser are based on the same model, the constructional information required to define either one is the same and both of these text files are structured identically.

Referring to Fig. 3, the first line consists of two integer fields. The first integer identifies whether the data that follows is for an evaporator (a value of '1') or for a condenser (a value of '2'). The second integer identifies the number of nodes that the heat exchanger is discretized into.

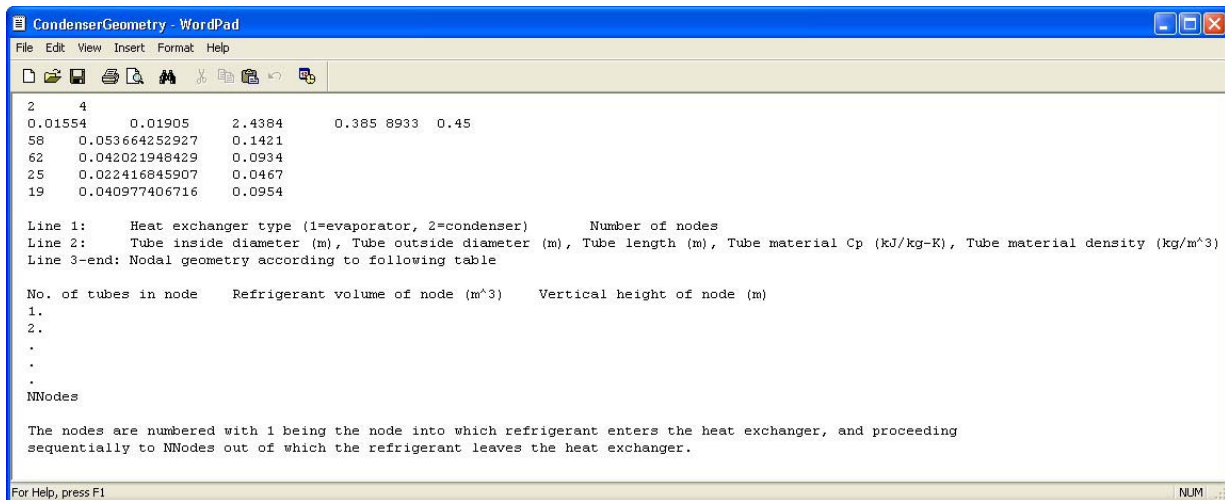


Fig. 3: Heat exchanger geometry specification format

The second line contains information about the tube size and material, listed in the order of inside diameter, outside diameter, length, specific heat and density. The sixth entry in this line is a fouling factor that will be described in the section on fault implementation.

All subsequent lines provide node specific information. The total number of lines must correspond to the number of nodes specified as the second integer in the first line. A mismatch will result in an error message and program termination. The lines are also to be arranged in the order that corresponds to the nominal flow-direction of the refrigerant through this heat exchanger with the first node being the one into which the refrigerant enters the heat exchanger and the last node as the one from which it leaves.

Each line of node information consists of three fields. The first field is the integer number of tubes encompassed in that node. The second field is the refrigerant volume in that node. The third field is the distance between the node faces in the vertical direction, i.e. in the direction of refrigerant flow.

SYSTEMGEOMETRY.TXT: This file lists the paths of all the other geometry text files and can be used to load different component details located in different directories.

The sub-directories *4NodeGeometries* and *MaxNodeGeometries* contain ready to use condenser and evaporator geometry files. The former apply to each heat exchanger

being discretized into 4 nodes. The latter apply to each heat exchanger being discretized into as many nodes as there are tube-rows in that heat exchanger (12 in case of the evaporator and 13 in case of the condenser).

The *IOFiles* sub-directory is where the input to and output from the model are stored. Fig. 4 shows the files within this sub-directory. The information required to initialize the system for start-up is stored in the text file(s) *Initial\_\*.txt*. The two possible initialization modes are designated *FULL* and *MINIMAL*. Full initialization consists of specifying the refrigerant pressure and refrigerant enthalpy<sup>1</sup> in both heat exchangers. Minimal initialization requires only three values, i.e., water temperatures leaving the evaporator and condenser, and total refrigerant charge in the system. It is to be noted that both of these modes of initialization pre-suppose that the system is in an equilibrium condition corresponding to the instant before start-up. This means that whichever mode is used to initialize the system, the controller begins with the compressor's inlet guide vanes in the minimum opening position and the RLA limit at its minimum.

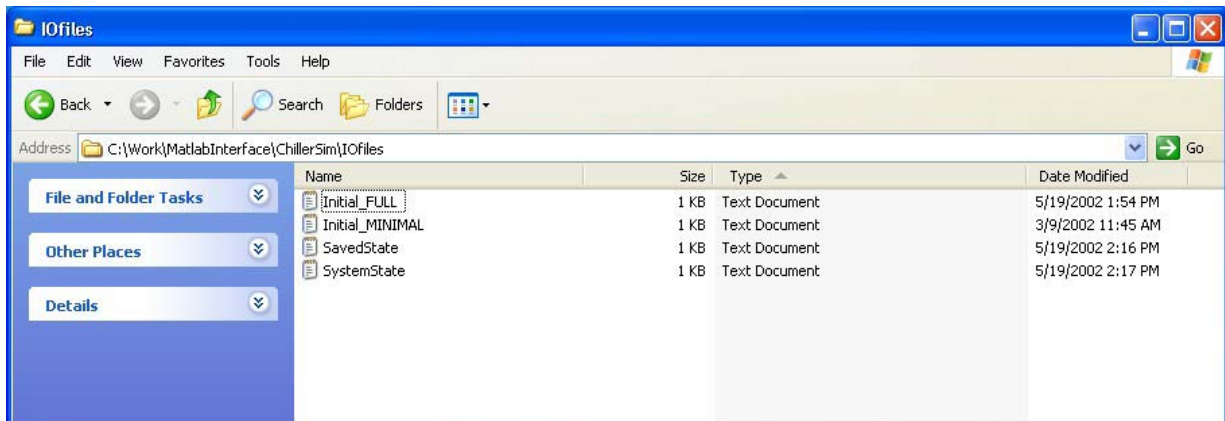


Fig. 4: Screen-shot of IOFiles directory

Once the chiller execution is begun, the controller gradually ramps up the limiting motor power thereby modeling the soft-start motor protection feature of the physical controller. In other words, the chiller system model can only be executed in a way that begins with a start-up. At the end of every 1s of simulation time, (i.e. the end of every loop of the

<sup>1</sup> Please see Appendix B for more information on refrigerant properties. The refrigerant enthalpy is referenced to 200 kJ/kg at 273.15K and the specific entropy to 1.00 kJ/kg-K at 273.15K, with the refrigerant in saturated liquid condition.

second nested level described in section 5.1 of the report) the state of the system is saved in the text file *SystemState.txt*.

There may exist a text file by name *SavedState.txt*. This file is created by the program when the user requires the current state to be saved between sessions. Saving and restoring the chiller state between sessions is described under the section on Usage. The following are the descriptions of the text files in the *IOFiles* directory.

INITIAL\_FULL.TXT: This is the file read by the chiller model when a full initialization is required. The format of information in this file is as shown in Fig. 5.

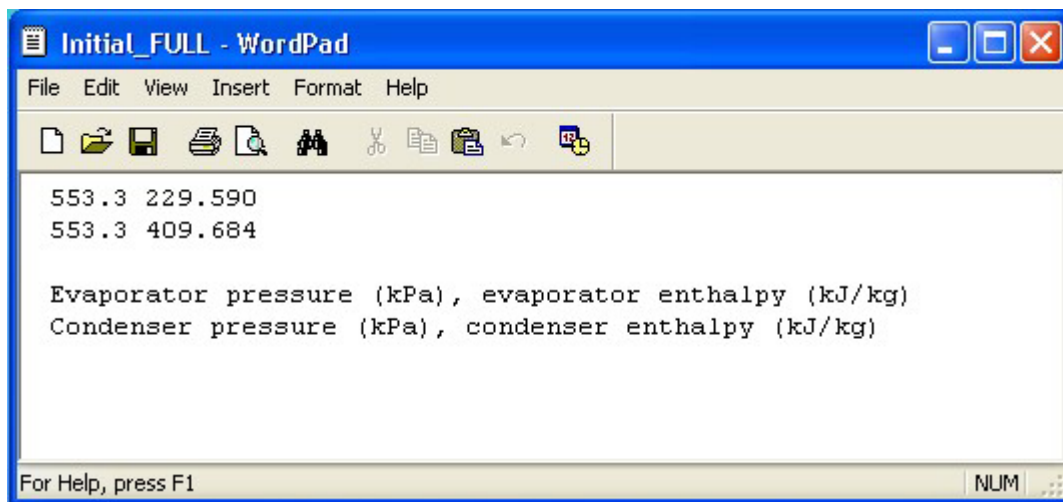


Fig. 5: Screen-shot of text file format for full initialization

INITIAL\_MINIMAL.TXT: This is the file read by the chiller model when a minimal initialization is required and is shown in Fig. 6.

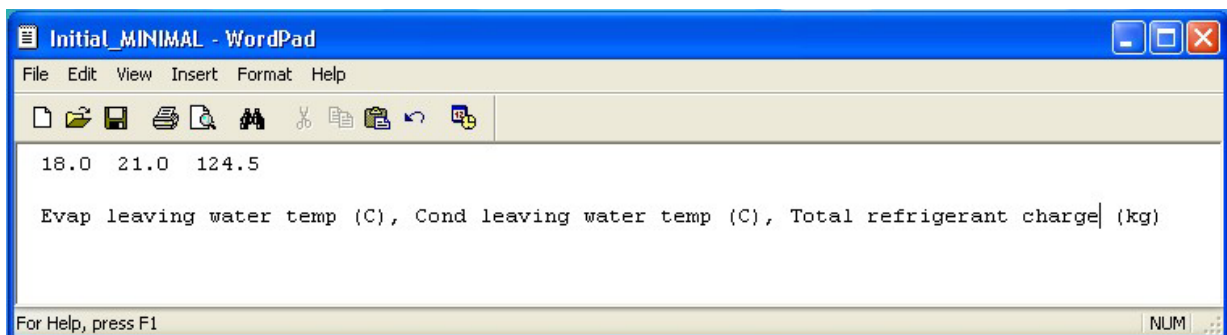
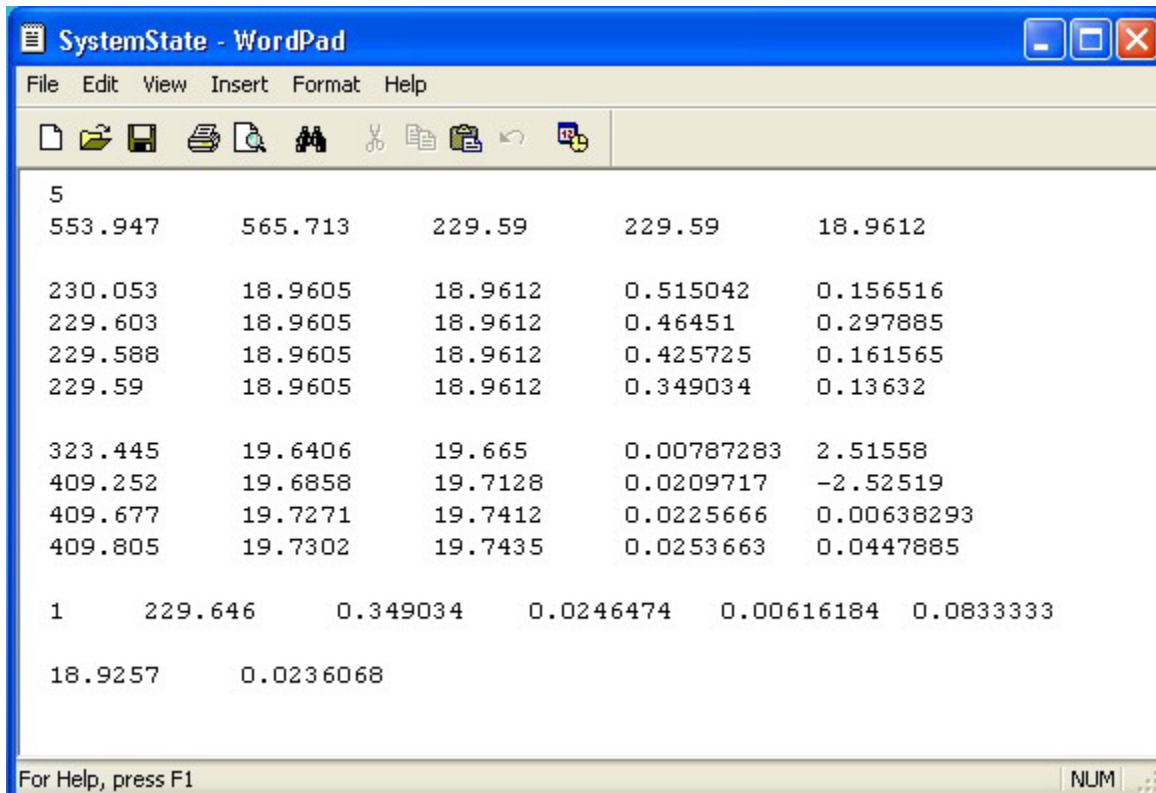


Fig. 6: Screen-shot of text file format for minimal initialization

SYSTEMSTATE.TXT: The system's states are saved in this file automatically every 1s of simulation in the format shown in Fig. 7.



5					
553.947	565.713	229.59	229.59	18.9612	
230.053	18.9605	18.9612	0.515042	0.156516	
229.603	18.9605	18.9612	0.46451	0.297885	
229.588	18.9605	18.9612	0.425725	0.161565	
229.59	18.9605	18.9612	0.349034	0.13632	
323.445	19.6406	19.665	0.00787283	2.51558	
409.252	19.6858	19.7128	0.0209717	-2.52519	
409.677	19.7271	19.7412	0.0225666	0.00638293	
409.805	19.7302	19.7435	0.0253663	0.0447885	
1	229.646	0.349034	0.0246474	0.00616184	0.0833333
18.9257	0.0236068				

Fig. 7: Screen-shot of text file format for system states information

Line1: simulation time i.e. number of seconds that the model has run since start-up

Line 2: evaporator pressure, condenser pressure, evaporator exit enthalpy, condenser exit enthalpy, chilled water temperature.

Evaporator nodal states (as many lines as there are nodes in the evaporator) in the following order:

nodal refrigerant enthalpy in kJ/kg;

nodal tube temperature in °C ;

nodal water temperature in °C ;

nodal refrigerant-side heat transfer rate in kW and

nodal refrigerant mass flow rate in kg/s.

Condenser nodal states (as many lines as there are nodes in the condenser) in the same format as above for the evaporator nodal states

Compressor state in the following order:

- controller operation mode (1 = start-up, 2 = normal);
- exit enthalpy in kJ/kg;
- mass flow rate in kg/s;
- motor power in kW;
- motor heat losses in kW and
- normalized guide-vane position ( $\gamma$ ).

Valve and bulb state in the following order:

- bulb temperature in °C and
- mass flow rate in kg/s.

The *SavedState.txt* file (if one exists) has a format identical to the *SystemState.txt* file above. The text files in the *Properties* directory are as described in Appendix B.

### Usage:

The chiller function can be called in Matlab with either one argument on the right-hand-side and none on the left, or with two arguments on the right-hand-side and two on the left, as below:

chiller(n)	(Single argument call)
y = chiller(t,u)	(Two-argument call)

### Single argument call:

When called with a single, integer argument the following actions are taken depending on the value of the integer argument:

- chiller(0) – performs a minimal initialization reading from *Initial\_MINIMAL.txt*.
- chiller(1) – performs a full initialization reading from *Initial\_FULL.txt*
- chiller(2) – saves the current state of the system to *SavedState.txt*.
- chiller(3) – loads the state of the system saved in *SavedState.txt*.

If a repeat initialization call is made, the existing chiller state information is simply overwritten. It is very important to note that all text filenames used by the program are unique and are overwritten without warning. Therefore, if any of the text files are desired by the user to remain unchanged, such files must be either renamed or relocated into another directory. This apparent lack of user-friendliness in fact allows for greater flexibility by allowing incorporation of the model into a program as a function call that is self-contained and that requires no real-time inputs from the user.

The chiller can also be initialized by reloading state information saved from an earlier session. This is done by entering 'chiller(3)' at the Matlab command prompt. On executing this, the text file *SavedState.txt* in the *IOFiles* directory is read and the system is restored to the state that existed when this text file was created. This file is created automatically by the program upon entering 'chiller(2)' at the command prompt. Any existing *SavedState.txt* file will be overwritten.

#### Two-argument-call:

After successful initialization, the chiller model can be executed by entering the following:

$$y = \text{chiller}(t,u)$$

't' and 'u' are the inputs required to drive the chiller model and 'y' is the output returned. The following is a description of these parameters.

t is an integer, positive number of seconds that the chiller is to be run;

u is the (5 x 1) vector of water-side boundary conditions in the order:

u[1] = Evaporator water entering temperature in °C

u[2] = Condenser water entering temperature in °C

u[3] = Chilled water set point temperature in °C

u[4] = Evaporator water mass flow-rate in kg/s

u[5] = Condenser water mass flow-rate in kg/s

y is a (29 x 1) vector of various system performance outputs, in the following order:

y[1] = Chiller simulation time since start-up in s

y[2] = Evaporator pressure in kPa



- y[3] = Condenser pressure in kPa
- y[4] = Refrigerant flow rate through compressor in kg/s
- y[5] = Refrigerant flow rate through valve only in kg/s
- y[6] = Refrigerant flow rate through cooling line only in kg/s
- y[7] = Sum of y[5] and y[6]
- y[8] = Motor power in kW
- y[9] = Motor heat losses in kW
- y[10] = Condenser water-side heat transfer rate in kW
- y[11] = Evaporator water-side heat transfer rate in kW
- y[12] = Evaporator leaving water temperature (chilled water temperature) in °C
- y[13] = Condenser leaving water temperature in °C
- y[14] = Superheat in °C
- y[15] = Sub-cooling in °C
- y[16] = Condenser refrigerant mass imbalance in kg
- y[17] = Evaporator refrigerant mass imbalance in kg
- y[18] = Energy balance across compressor in kW
- y[19] = Energy balance across condenser in kW
- y[20] = Energy balance across evaporator in kW
- y[21] = Refrigerant specific enthalpy leaving evaporator in kJ/kg
- y[22] = Refrigerant specific enthalpy leaving compressor in kJ/kg
- y[23] = Refrigerant specific enthalpy leaving condenser in kJ/kg
- y[24] = Refrigerant specific enthalpy entering evaporator in kJ/kg
- y[25] = Valve lift in m
- y[26] = Valve flow area in m<sup>2</sup>
- y[27] = Refrigerant mass in condenser in kg
- y[28] = Refrigerant mass in evaporator in kg
- y[29] = Total refrigerant mass in the system in kg

The usage of the model is illustrated by a series of examples described below and included as m-files with the software. These m-files can be used as templates by the user.

**Example 1 (*Ex1.m*): Start-up in fault-free condition:**

This example demonstrates the preparatory steps that precede execution of the model, followed by the actual execution of the chiller model through the start-up. When the steady-state is reached, the execution is stopped and the state of the system at that time is saved for future use.

Step 1- System Definition: The default heat-exchanger geometry is used, i.e., as defined in the files in the *Geometries* directory.

Step 2 – Initialization: The default initialization of *Initial\_FULLL.txt* is used.

Step 3 – Boundary conditions: The  $u$  vector is defined for the start-up period. For simplicity, it is assumed to remain constant during the complete start-up region. The desired set point is 10°C. The normal water-flow rates of 13.2kg/s in the evaporator loop and 16.7kg/s in the condenser water loop are used. The evaporator return water temperature is 16°C and the condenser return water temperature is 30°C.

Referring to the code in *Ex1.m*, (Fig. 8) line 4 is the *FULL* initialization step. This is followed by the setting of the water-side boundary conditions. The chiller output plotting rate is specified in line 29. With this information, the chiller execution loop is begun at line 34. For certain combinations of initial conditions and entering water temperature change rates during early (<150s) start up, it has been found that the model fails to converge. A full characterization of this numerical issue is in progress, but it can be overcome by gradually and linearly ramping the entering water temperature from the initial condition to the final value over the first 120s-150s. This is shown in lines 39-43.

Line 44 updates the input vector  $u$  and is followed by the execution of the chiller through a 10s loop. The output of the chiller is recorded at every 1s and saved in the *output* array which is saved to the disk every 10s (line 57). The 0.1s pause at line 59 is required only to allow the figure plots to refresh. The final state of the chiller is saved (line 62) into *SavedState.txt* for future use. Fig. 9 shows selected output of this example.

```

c:\work\MatlabInterface\ChillerSim\Ex1.m
File Edit View Text Debug Breakpoints Web Window Help
[Icons] Stack: Base

1 %Chiller Model Examples
2 %Example 1
3 %
4 chiller(1);
5
6 %Define water-side boundary conditions
7 TEWI = 16;
8 TCWI = 30;
9 TEWO_SET = 10;
10 MEWAT = 13.2;
11 MCWAT = 16.7;
12
13 %Set-up plotting
14 FIG = figure;
15 set(FIG,'Position',[231 132 1128 908]);
16 subplot(311); axis([0 1000 0 1500]);
17 grid on; hold on;
18 xlabel('s'); ylabel('kPa');
19 subplot(312); axis([0 1000 0 50]);
20 grid on; hold on;
21 xlabel('s'); ylabel('deg C');
22 subplot(313); axis([0 1000 0 100]);
23 grid on; hold on;
24 xlabel('s'); ylabel('kW');
25
26 %Execute chiller for 1000s
27 %
28 %Update water side conditions every 10s
29 t = 10;
30 %Initialize counter and output-storage
31 i = 1;
32 output = [];
33 %Begin loop...
34 while(i<1000)
35     Twi = TEWI;
36     Tewo_set = TEWO_SET;
37     mewat = MEWAT;
38     mcwat = MCWAT;
39     if(i<120)
40         Twi = 21 + i*(TCWI-21)/120;
41     else
42         Twi = TCWI;
43     end
44     u = [Twi;Twi;Tewo_set;mewat;mcwat];
45     j = 1;
46     while(j<=t)
47         y = chiller(1,u);
48         output = [output;y'];
49         j = j + 1;
50     end
51     subplot(311);
52     plot(i,y(2),'b.',i,y(3),'r. ');
53     subplot(312);
54     plot(i,y(12),'b.',i,y(13),'r. ');
55     subplot(313);
56     plot(i,y(8),'r. ');
57     save output;
58     i = i+10;
59     pause(0.1);
60 end
61 %Save state at the end
62 chiller(2);

```

Ready

Fig. 8: m-code of *Example 1*.

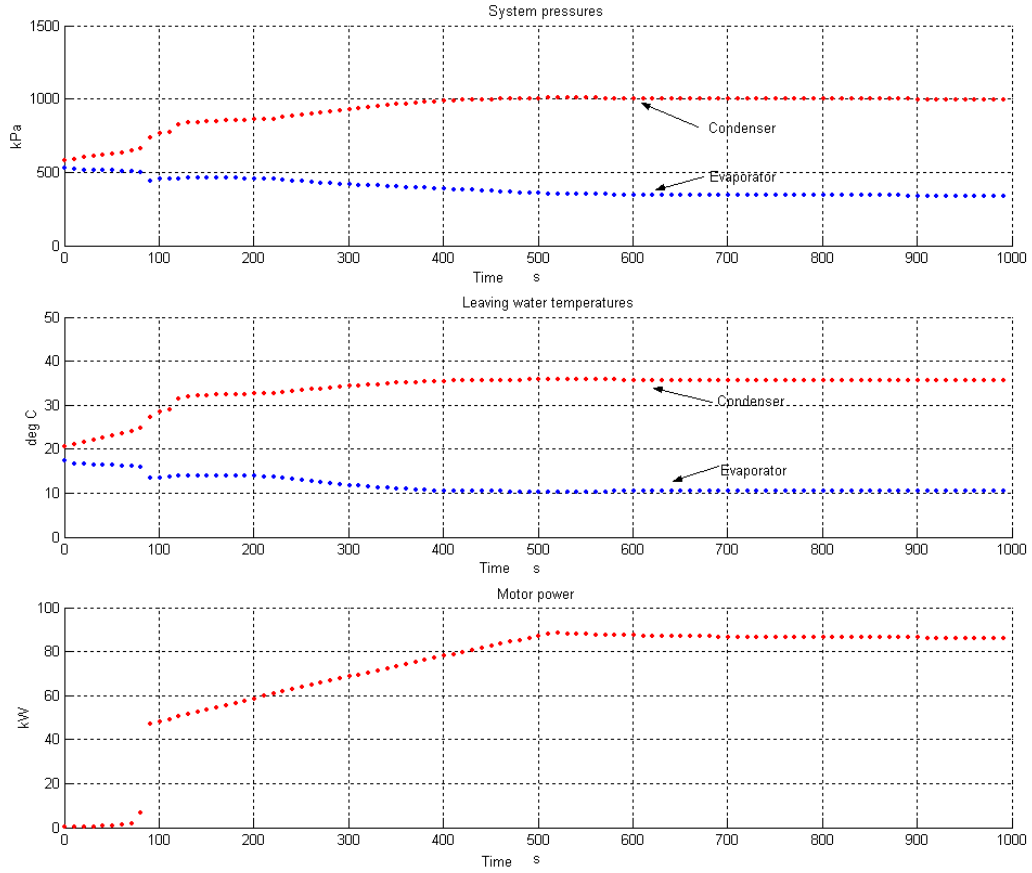


Fig. 9: Output plot of selected chiller parameters in *Example 1*.

During the first 90s of the simulation the model uses a fixed value of polytropic efficiency for the compressor because the map (eqn. 20) does not apply during that time. This results in a low compressor flow-rate and therefore a low power prediction. This also slows down the early response of the pressures and water temperatures. Once the efficiency map becomes applicable (at 90s), the system's response is seen to change significantly. Thereafter, the solution proceeds smoothly to the steady state. The linear power variation up to ~500s is the effect of the current limit imposed by the controller which prevents the compressor from delivering large flow-rates to rectify the large initial chilled water temperature error.

**Example 2 (Ex2.m): Evaporator water entering temperature change in fault-free condition**

This example demonstrates the revival of a system state from an earlier saved state<sup>2</sup>, followed by executing the system model by driving it through a transient triggered by a 2°C step drop in the evaporator return water temperature. The step-change occurs 50s after the start of the execution. When steady-state is reached again (150s later), the execution stops and the system state is saved. Fig. 11 shows the m-code for this example. The significant differences are the initialization, which now is done by loading the by loading the chiller state from the earlier saved state (line 4), the way the boundary conditions are updated in lines 40-43 and the boundary condition updation and result plotting sampling time (line 29) which is now done every 2s. Fig. 10 shows the output for this example.

---

<sup>2</sup> It is assumed here that the text file *SavedState.txt* is as was saved at the end of *Example 1*. If this is not the case, move the *SavedState.txt* file (if one exists) to another location on the disk, copy the file *SavedState\_1000.txt* from the *IOFiles* directory to the *Chillersim* directory and rename it *SavedState.txt*.

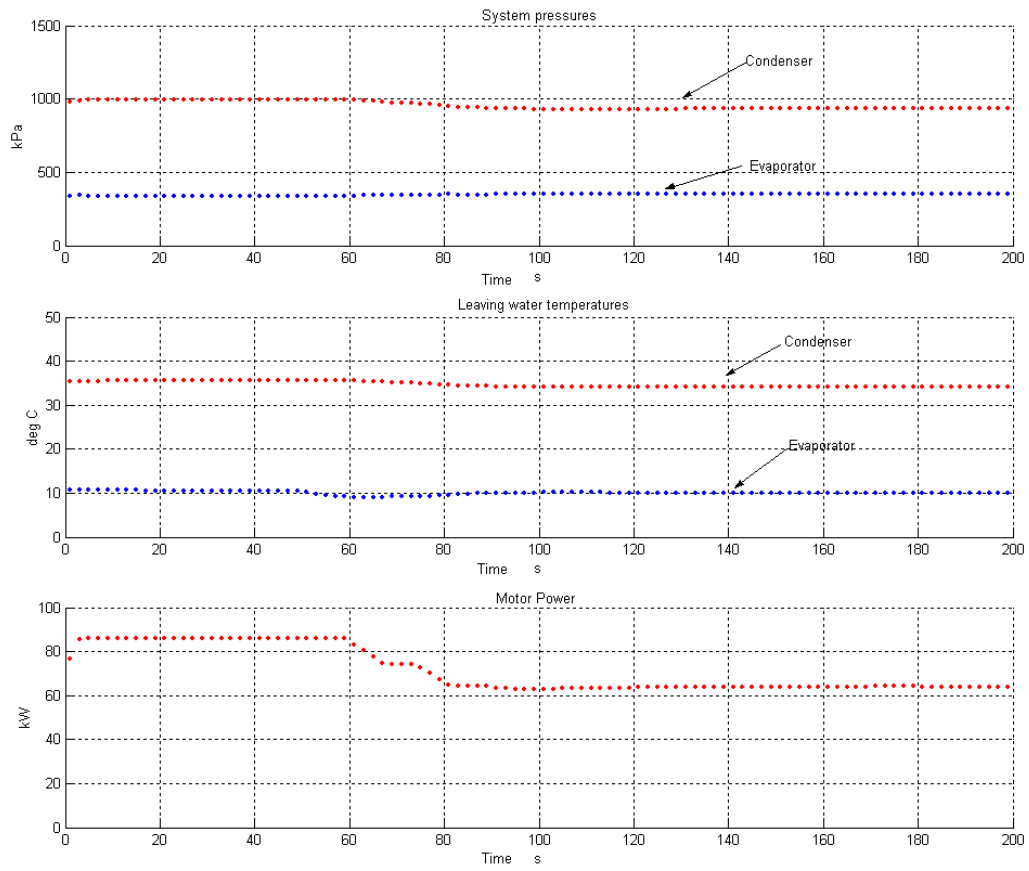


Fig. 10: Output plot of selected chiller parameters in *Example 2*.

```

1 %Chiller Model Examples
2 %Example 2
3 %Load chiller state
4 chiller(3);
5
6 %Define water-side boundary conditions
7 TEWI = 16;
8 TCWI = 30;
9 TEWO_SET = 10;
10 MEWAT = 13.2;
11 MCWAT = 16.7;
12
13 %Set-up plotting
14 FIG = figure;
15 set(FIG,'Position',[231 132 1128 908]);
16 subplot(311); axis([0 200 0 1500]);
17 grid on; hold on;
18 xlabel('s'); ylabel('kPa');
19 subplot(312); axis([0 200 0 50]);
20 grid on; hold on;
21 xlabel('s'); ylabel('deg C');
22 subplot(313); axis([0 200 0 100]);
23 grid on; hold on;
24 xlabel('s'); ylabel('kW');
25
26 %Execute chiller for 200s
27 %
28 %Update water side conditions every 2s
29 t = 2;
30 %Initialize counter and output-storage
31 i = 1;
32 output = [];
33 %Begin loop...
34 while(i<200)
35     Tewi = TEWI;
36     Tcwi = TCWI;
37     Tewo_set= TEWO_SET;
38     mewat = MEWAT;
39     mcwat = MCWAT;
40     if(i>=50)
41         Tewi = TEWI - 2;
42     end
43     u = [Tewi;Tcwi;Tewo_set;mewat;mcwat];
44     j = 1;
45     while(j<=t)
46         y = chiller(1,u);
47         output = [output;y'];
48         j = j + 1;
49     end
50     subplot(311);
51     plot(i,y(2),'b.',i,y(3),'r. ');
52     subplot(312);
53     plot(i,y(12),'b.',i,y(13),'r. ');
54     subplot(313);
55     plot(i,y(8),'r. ');
56     save output;
57     i = i+t;
58     pause(0.1);
59 end
60 %Save state at the end
61 chiller(2);

```

Fig. 11: m-code of *Example 2*

A drop in evaporator entering water temperature, keeping the same chilled water set-point and water-flow rate, corresponds to a drop in the building load. This results in lesser heat transfer to the refrigerant in the evaporator. This causes the leaving water temperature to drop below the set-point thus far maintained, as seen in Fig. 10. The reduced evaporator capacity implies that the motor now has to deliver lesser power and the condenser has to reject lesser heat to the cooling water. The stair-step reduction in motor power is caused by the step-and-wait action of the controller that now responds to the negative error in chilled water temperature. The reduced compressor flow rate and necessary condenser heat duty result in the lower condenser pressure and condenser leaving water temperature.

**Example 3 (*Ex3.m*):** Evaporator and condenser water entering temperature change in fault-free condition:

This example is a repeat of Example 2, with the difference that the condenser entering water temperature is also changed (increased) by 2°C during the transient triggered by a 2°C drop in evaporator return water temperature. The m-code can be seen in *Ex3.m*. Fig. 12 shows the output for this combination transient.

The increased condenser water temperature results in a higher (than in Example 2), condenser pressure and motor power. The higher condenser pressure is caused by the need to sustain the temperature difference between the refrigerant and (the now warmer) water in the condenser which will allow the required heat transfer rate. The stabilized condenser leaving water temperature is also seen to be higher as a result. The motor power is higher because of the increased pressure difference that the compressor now needs to work against.

The condenser leaving water temperature is seen to first drop, in response to the reduced evaporator entering water temperature, and then increase because of the increase in the condenser entering water temperature. The chilled water temperature also responds similarly but the controller manages to return it to the un-changed set-point.



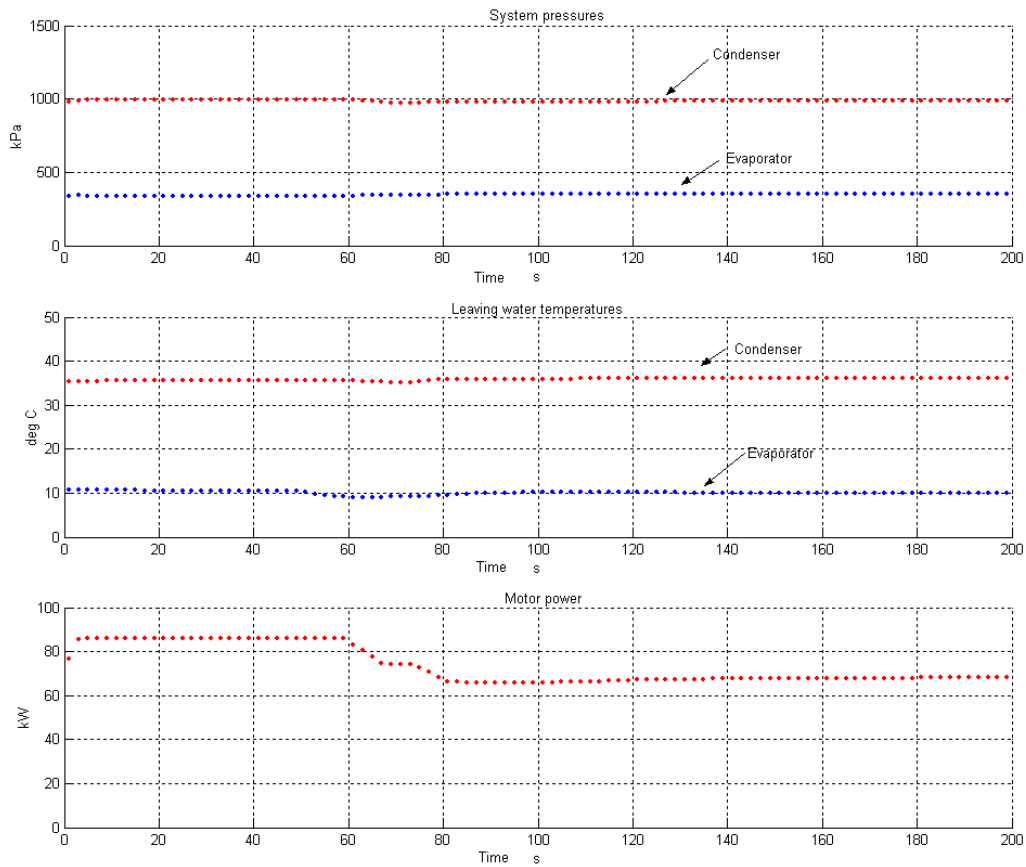


Fig. 12: Output plot of selected chiller parameters in *Example 3*.

**Example 4(Ex4.m): Evaporator and condenser water entering temperature change and chilled water set-point temperature change in fault-free condition:**

This example includes a 2°C increase in the chilled water set-point temperature into the boundary conditions imposed on Example 3. The system begins with the same initial condition as exists at the end of Example 1 (see footnote on pg. 13). Fig. 13 shows the output. Note the large drop in motor power with the reduced load caused by a smaller evaporator return water temperature as well as increased set point temperature. The chilled water temperature settles down to the new set point stably. The evaporator pressure shows some small scale transients between 100 and 120s. This is caused by the dynamics in the valve.

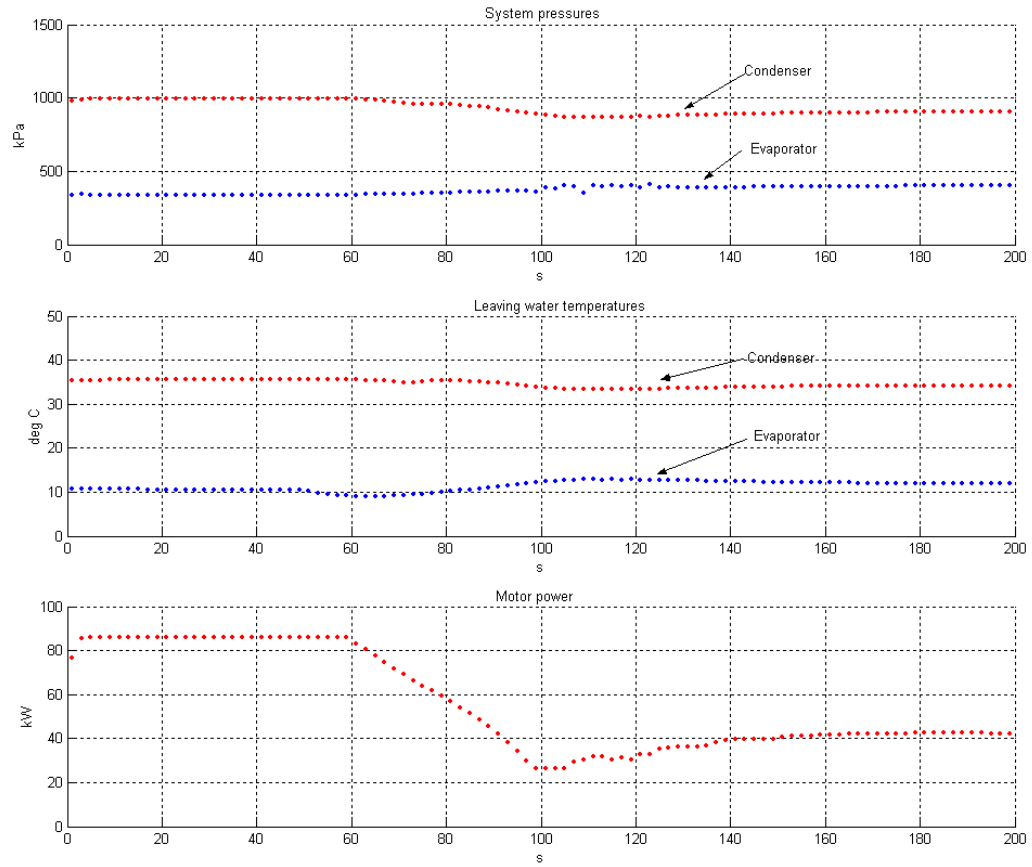


Fig. 13: Output plot of selected chiller parameters in *Example 4*

### Fault simulation

The preceding examples demonstrate the use of the model and its output for the fault-free condition. This section demonstrates how the system performance can be generated with faults introduced. The following faults can be emulated in the system model:

- (a) Up to 40% reduction in water flow-rates in one or both heat-exchangers
- (b) Up to 20% refrigerant undercharge in the system
- (c) Up to 20% refrigerant overcharge in the system
- (d) Up to 45% fouling in one or both heat-exchangers

### Example 5 (*Ex5.m*) Reduced water flow rates:

This fault can be introduced simply by altering the values entered in the input vector  $u$  above. This fault can be introduced either as a fully-developed one or as a gradually developing one. The nominal water flow-rates are 13.2 kg/s in the evaporator water loop and 16.7 kg/s in the condenser water loop. This example demonstrates the reduction in condenser flow-rate as a gradually developing fault. Water flow-rate reduction in the evaporator loop, or in both evaporator and condenser water loops, can also be implemented in the same manner.

The system begins at a state as at the end of Example 1. At this point in time, the condenser water flow rate is at its normal value. Ten seconds after the execution starts the fault begins to develop. Over the next 180s the flow rate drops linearly to 60% of the normal, i.e., 10 kg/s. Fig. 14 shows the output of this simulation.

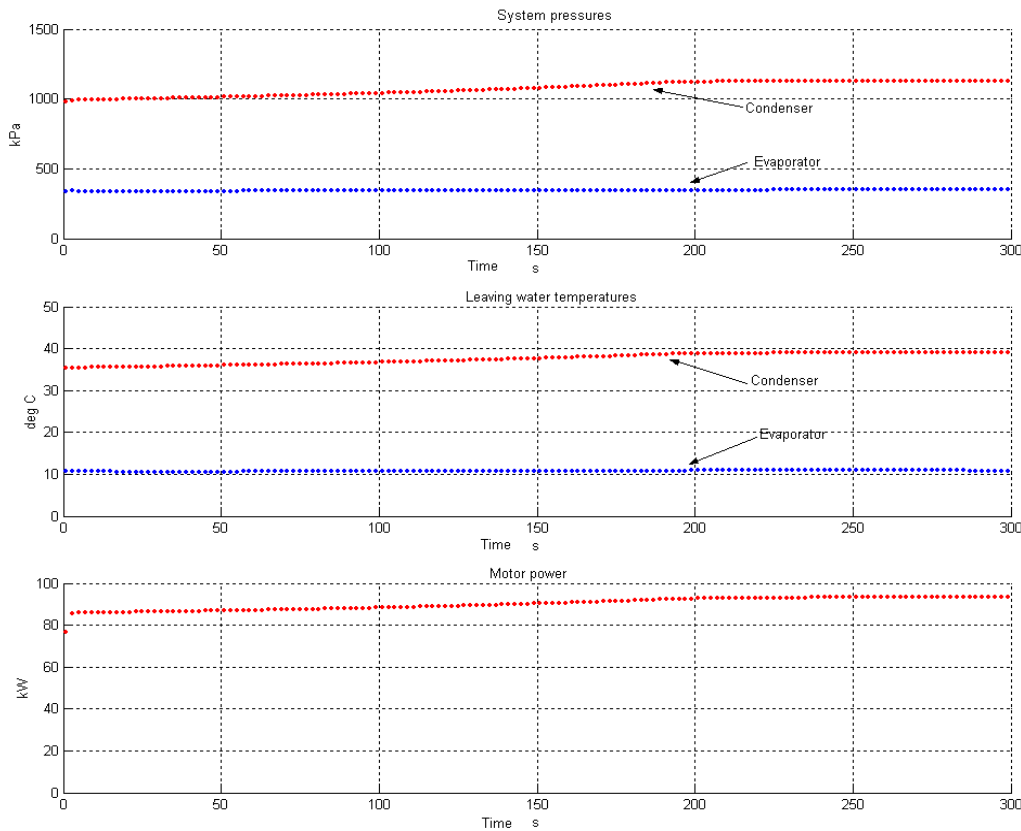


Fig. 14: Output plot of selected chiller parameters in *Example 5*

As expected, the reduced water flow rate results in lesser heat transfer rate on the water side in the condenser. This necessitates a higher temperature difference between the refrigerant and the water, which is achieved by a higher condenser pressure. The evaporator pressure is virtually unchanged. The motor power increases because of the higher pressure difference across the compressor.

**Example 6 (Ex6.m): Charge variation:**

Refrigerant undercharge and overcharge are implemented in the same manner, i.e., through the initial enthalpy distribution when using a full initialization or by simply specifying the total refrigerant charge when using a minimal initialization. The refrigerant charge quantity normal for the system model is 124.7 kg (see section **Error! Reference source not found.**). Varying the magnitude of these faults while the model executes is not incorporated in the present code.

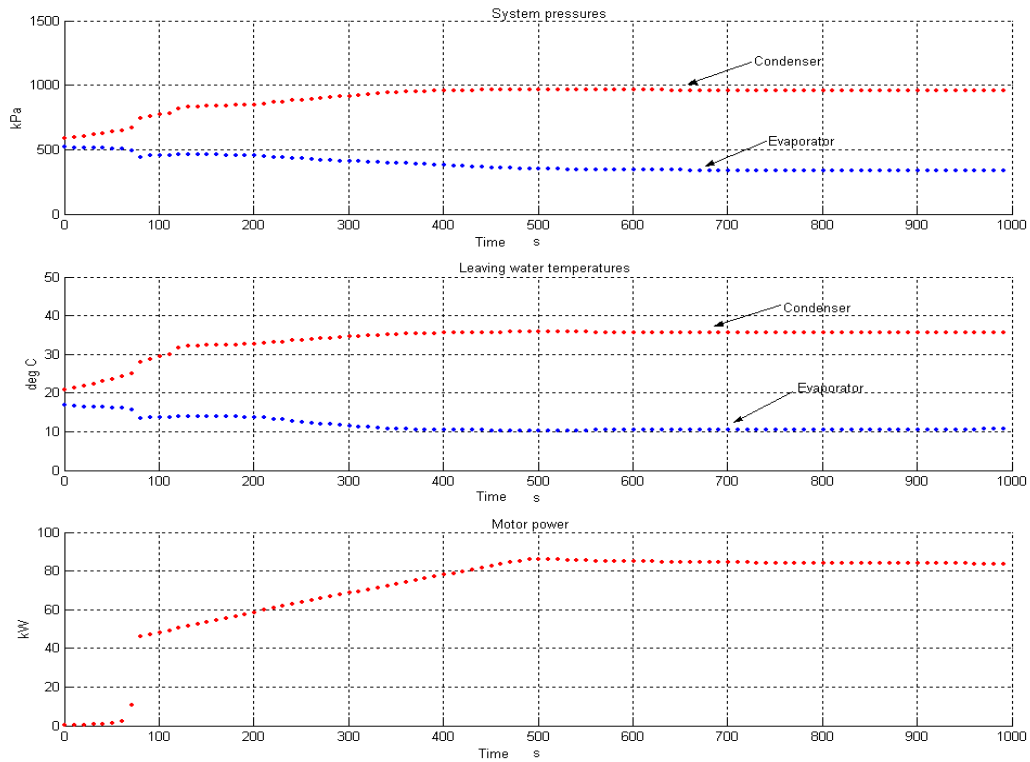


Fig. 15: Output plot of selected chiller parameters in *Example 6*

In this example, the system is initially charged with 80% of the nominal charge, i.e., with 99.76 kg of refrigerant. This is done by altering the third field in *Initial\_MINIMAL.txt*. The model is run from start-up through to the achievement of steady-state. The m-code for executing this example is identical to that of *Example 1*. Fig. 15 shows the output. In comparison with the output of *Example 1* which is with the correct refrigerant charge, the system pressures are seen to be lower as is the motor power. Other parameters can be plotted by loading *output.mat* into the Matlab workspace. The columns of *output* are indexed identically to the output vector *y* listed earlier in this section.

Refrigerant overcharge can be similarly implemented by setting the value in *Initial\_MINIMAL.txt*. Alternatively, for either case of charge variation, the enthalpy distribution can be set in *Initial\_FULL.txt* and the chiller can be initialized by a single-argument call with a value of 1. The refrigerant volumes of the heat exchangers can be determined from their geometry files.

#### **Example 7: Heat exchanger fouling:**

Heat exchanger fouling is implemented by specifying fouling as a percentage loss in heat transfer conductance or area. The fouling applies only to the water-side of the tube. This fault parameter is entered along with the heat exchanger geometry, as the sixth field of the second line in the geometry file shown circled in Fig. 16. The magnitude of fouling cannot be altered during model execution in the present code. Any of the example m-codes can be used with this change made in the geometry file.

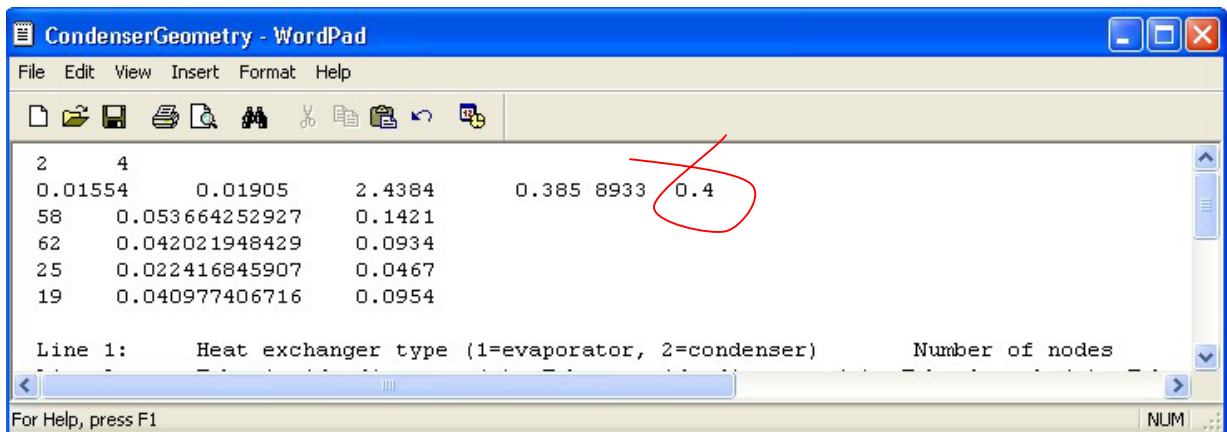


Fig. 16: Screen-shot of condenser geometry file with 40% fouling introduced

**Example 8 (Ex8.m): Boundary condition updation from file:**

The final example m-file is the file used to execute the chiller model through the data presented under the section on validation. This file demonstrates how the boundary conditions stored in a text file on disk can be read at pre-determined intervals and used to execute the chiller model. The user is encouraged to examine the code in this m-file as much of it is self-explanatory.

**Error handling and reporting**

Error messages generated are broadly divided into those that are launched from within the Matlab interface and those that are launched from within the model code. The errors trapped within the model code are logged to a text file *error1.log*, which is created automatically when the chiller model is run for the first time in any session. The Matlab interface generated errors consist of the following:

1. 'Invalid call to chiller routine' – the number of arguments passed and returned in the call to chiller does not match either the single-argument call format or the two-argument call format described above.
2. 'Cannot create engine' – a chiller model could not be created in memory, likely because of insufficient memory.
3. 'System construction failed' – defining the system components geometry failed. This can occur if the paths to the geometry files have not been included into Matlab's search path or if any of the geometry files are of incorrect format. In case of the latter, additional error information is logged to *error1.log*.
4. 'Initialization code needs to be an integer' – the value passed in a single-argument call to chiller was not an integer. Passing anything other than an integer will trigger this error.
5. 'Invalid argument value' – the value passed in a single-argument call to chiller was not any of 0,1,2 or 3. These are the only acceptable values for a single-argument call.
6. 'System state could not be saved' – saving the system state in the text file *SavedState.txt* failed. A possible cause of this is insufficient disk space.

7. 'System state could not be loaded' – loading a previously saved system state failed. Possible causes of this are:
  - a. the file *SavedState.txt* was not found because the path was not included into Matlab's search path
  - b. the format of information in the file *SavedState.txt* is incorrect. Additional error information is logged in *error1.log*.
  - c. one or more of the system components could not be initialized with the data in *SavedState.txt*. Additional error information is logged in *error1.log*.
8. 'System initialization failed' – the system components could not be initialized with the data in *Initial\_FULLL.txt*. Possible causes are:
  - a. the file *Initial\_FULLL.txt* was not found because the path was not included into Matlab's search path
  - b. the format of information in the file *Initial\_FULLL.txt* is incorrect. Additional error information is logged in *error1.log*.
  - c. one or more of the system components could not be initialized with the data in *Initial\_FULLL.txt*. Additional error information is logged in *error1.log*.
9. 'Chiller not initialized' – an attempt was made to use the chiller model without performing an initialization.
10. 'Chillersim error. Consult error log file for details' – an error occurred in the model code and additional error information has been logged in *error1.log*.

Fig. 17 shows an example of error information that is logged into *error1.log*. The error entry shown is that of an error in the format of information provided in the initialization file. The error messages are logged from local to global, i.e. the first error message identifies the location in the code where the effect of the error was observed. Subsequent error messages help identify the path through the code that was being executed at the time the error occurred. Since the cause of the error can lie away from the point where its effect is observed, the error message can only be generic and a full listing of these error messages serves little purpose.

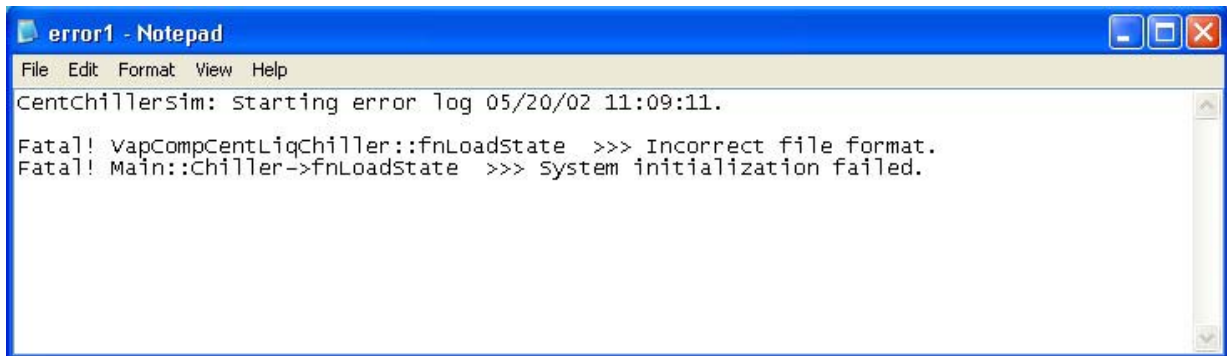


Fig. 17: Screen-shot of error log file with an example error entry

The error trapping structure has been designed to trap most of the common errors that were conceived as possible. As usage of the model increases, additional information will be available from users that will help identify errors and bugs not yet detected and the code can be made more robust.